Servidor de busca aberto

Busca com qualidade

Analisamos a ferramenta de indexação Open Search Server e mostramos como integrar o recurso de busca ao seu website utilizando PHP.

por Markus Feilner, Thomas Pfeiffer e Markus Heller

A *startup* (empresa de tecnologia recém criada) francesa Jaeksoft foi criada em fevereiro de 2010. Apenas um anos depois, o fabricante lançou a versão 1.2 de sua ferramenta de busca e indexação, o *Open Search Server* [1]. Sua relativa imaturidade é evidente, assim como alguns aborrecimentos. Mas também há uma quantide de recursos impressionantes.

Indexação livre

De acordo com seu fornecedor, o Open Search Server (OSS) vai "vasculhar sistemas de arquivos, bancos de dados e sites para criar rapidamente índices confiáveis e dar suporte a um processo preciso de investigação". Em outras palavras, você pode usar o OSS para adicionar funções de busca a um website ou outro banco de dados e pode até construir uma máquina de busca agregada para indexar e buscar dados de múltiplas fontes na Internet. O indexador OSS suporta uma longa lista de formatos de arquivos. Uma API está disponível para acesso rápido e sem complicação a resultados e a maioria de seus muitos recursos (**figura 1**).

O alicerce da ferramenta é o Java em um servidor Tomcat [2]. O mecanismo de busca Lucene [3] permite que administradores e desenvolvedores usem sua sintaxe de interpretador de consultas e funções de classificação fora do gerenciamento da interface (**quadro 1**). O OSS depende do Quartz [4] como agendador. A interface, rápida e flexível, é implementada em Zkoss [5] e funciona com qualquer navegador compatível com Ajax.

Figura 1 Lucene, Tomcat e uma API flexível combinados no Open Research Server.

Suporte corporativo

Além da variante livre, lançada sob muitas licenças (incluindo AGPL e GPLv3) por conta de sua estrutura modular, o fornecedor, a Jaeksoft, também oferece uma versão corporativa, com suporte opcional, acordo de nível de serviço (Service Level Agreement - SLA) e recurso avançado na forma de módulos instaláveis. O fornecedor também oferece desenvolvimento, implementação e otimização dos módulos adicionais.

A versão *Community* é facil de instalar: faça o download do arquivo `tar.gz` do site, abra-o em um servidor web e execute `start.sh` no shell (`start.bat` no Windows). São necessários um servidor de web em funcionamento com PHP5, `php5-curl` e *Java Runtime Environment*.

O pacote não inclui um script *init*; se você deseja executar o servidor permanentemente, é importante prestar atenção no manual. Infelizmente, o wiki do OSS [6] não oferece mais do que detalhes sobre o procedimento de instalação. Para esse artigo, usamos a versão `open-search-server-1.2.1-r987.tar.gz` de 30 de março de 2011.

Primeiro contato

A pasta `apache-tomcat-6.0.32` no pacote compactado contém alguns caracteres familiares: o aplicativo do servidor Tomcat vem pré-configurado com seus binários, arquivos de log e de configuração. Se algo não funciona, cheque o arquivo de log `catalina.out`. O Tomcat oferece dicas úteis caso algo não funcione em sua instalação. Os índices ficam no diretório `data`, onde você também encontrará um subdiretório com a configuração em arquivos HTML. Você pode usar um script ou adicionar URLs e padrões aqui, mas vai precisar reiniciar o OSS depois de fazer isso.

A interface web

Depois de iniciar, o Tomcat escuta a porta padrão 8080. Você deve manter essa porta atrás do firewall para evitar acessos não autorizados e deixá-la disponível apenas para serviços locais. Se você usar seu nvegador para acessar a URL `http://localhost:8080//`, será levado à interface de gerenciamento, que inicialmente apenas mostra as abas *Index* e *Privileges*. Você pode começar criando um novo índice ou usuário com privilégios. Modelos estão disponíveis para índices (*índice vazio*, *buscador de arquivo* e *buscador web*). Depois, você pode selecionar um índice para editar. Uma vez carregado o índice, marcações de tempo são adicionadas ao menu do OSS, como na **figura 2**.

A falta de documentação pode fazer com que as abas, opções de configuração e itens secundários sejam um pouco confusos. Entretanto, os modelos (*templates*) e funções autoexplicativas são fáceis de serem iniciados. O atributo *Schema* permite a edição de detalhes nas funções de busca; *Query* permite a formulação de buscas ou coordena o formato do resultado de saída. O OSS pode confrontar os resultados (*Query* > *Collapsing*) caso critérios específicos permitam a conclusão de que sites idênticos existem.

Os modos disponíveis aqui são `full`, `cluster` e `optimized`. Por causa da falta de documentação, a única forma de descobrir qual modo é melhor para o seu projeto é experimentando. *Snippet fields* (também presente na *Query tab*) define o formato de saída. Você pode definir os campos e seus

Quadro 1: Lucene

O coração de qualquer mecanismo de busca moderno é o indexador. O indexador escalonável criado usando a biblioteca livre do Lucene Java trabalha bem se comparado com mecanismos de busca comerciais.

Analisador, Indexador e Fields

O princípio fundamental é simples: um analisador separa um documento e distribui os metadados deste e componentes de texto por campos predefinidos. Então, o criador de índice cria o índice em si. O criador de índice é tipicamente um transdutor automático, que leva você a uma lista de links quando é introduzido um termo. Especialistas referem-se a isto como "arquivos invertidos", porque um índice corresponde mais ou menos ao trabalho original. Entretanto, os trabalhos não são mais armazenados na sua ordem natural, mas sim em ordem alfabética com um apontador para suas posições originais.

Criar e manter a máquina em funcionamento e garantir armazenamento compacto dos link no arquivo do sistema normalmente é uma tarefa arriscada que o Lucene realiza de uma forma muito elegante. Ele facilmente alcança um desempenho de indexação de 30MBps em uma máquina moderna, enquanto reduz o texto original em torno de 20% no índice. Obviamente, esse processo depende da extensão para a qual o mecanismo de busca acessa o documento e o distribui pelos campos.

Boost e Page Rank

Um programador pode usar os campos oferecidos para dar, por exemplo, uma atenção especial aos títulos, dando mais peso a esse elemento, ou usando metadados externos, como nome ou autor do arquivo. Se o termo de busca ocorre nestes campos, o elemento indexado recebe um valor de ranking mais alto do que se apenas ocorresse em algum lugar no corpo do texto. O levantamento correto destes campos vai levar ao tipo de ranking que os usuários querem encontrar. O Lucene pode também analisar campos individuais de uma forma focada ou usá-los para busca de resultados.

Além das análises booleanas relativamente entediantes, você pode também realizar buscas com coringas ou tolerantes a erros. O objetivo aqui é não apenas encontrar resultados com prefixo idêntico (como em SQL com `'Linux Mag%'`), mas também com grafias diferentes. Graças ao método *Levenshtein Distance*, você pode definir sua própria sintaxe para descobrir todas as variáveis ortográficas do `Schmie?[dt]` + (Schmid/Schmit/Schmidt, com ou sem um `e` depois do `i`) ou `M[ae][iy]e?r` (Meyer/Mayer/Maier/etc.) sem comprometer o desempenho.

Mantenha distância!

Qualquer tipo de análise a distância recebe suporte do operador de proximidade (~) em análises com múltiplas palavras. Ele encontra todos os tipos de seções de textos nos quais várias palavras ocorrem até a distância máxima de uma para a outra. Em contraste com documentos estáticos ou levantamento de campos, isso significa que você pode levantar mais termos que os outros durante o tempo de execução. O termo operador de *boost* (^) também permite que você interprete chaves de busca individuais como mais importantes do que outras com consultas de múltiplas palavras.

Resumindo, usuários típicos não precisarão se preocupar com estas coisas em um mecanismo de busca ideal, porque os resultados mais relevantes estarão no topo da lista de qualquer forma – mesmo que não seja utilizado operadores. O indexador do Lucene inclui um grande número de ferramentas para lidar com isso.

tamanhos aqui e, então, configurar a lista de resultados.

A aba *Crawler* é onde você especifica o alvo de sua busca: websites, bancos de dados ou sistemas de arquivos. Vários filtros estão disponíveis para auxiliá-lo a modificar isso. A **figura 2** mostra a subjanela do *Crawl process* para uma máquina que está indexando o site inteiro da **Linux Magazine**.

Uma das configurações pode ser mal interpretada: se você for levado a achar que precisa aumentar ao máximo o número de sites por *host* para um servidor web com grande quantidade de conteúdo, vai provavelmente cair em uma armadilha. O valor somente diz ao OSS quantas páginas recuperar na sessão antes de adicioná-las em seu índice. Se você configurar este valor muito alto, usará memória RAM demais ou possivelmente verá a mensagem de erro ilustrada na **figura 3**. Em muitos casos, o OSS ou Java congela quando não tem mais memória.

Se você quer criar um índice de um site grande, certifique-se de que tenha memória suficiente em seu sistema. Para alocar 2 GB de RAM no Java Virtual Machine, você simplesmente precisa adicionar as duas linhas seguintes ao seu script de inicialização:

```
CATALINA_OPTS="-Xms2G -Xmx2G
➥ -server"
export CATALINA_OPTS
```

É possível usar mais do que 2 GB em um sistema operacional de 64 bits com Java de 64 bits. Neste caso, as opções a seguir permitem 6 GB de memória RAM para o Java:

```
CATALINA_OPTS="-d64 -Xms6G -Xmx6G
➥ -server"
export CATALINA_OPTS
```

Ferramentas de monitoramento, tais como Darkstat [7], nos repositórios Debian, monitoram o tráfego que isso cria ao mesmo tempo (**figura 4**).

Alguns firewalls ou servidores web classificam ondas de consultas muito frequentes como um ataque de negação de serviço (DoS) e bloqueiam o cliente.

De acordo com o fornecedor, o OSS é capaz de indexar 16 linguagens; pode identificar palavras inteiras, fragmentos, ou variantes de uma forma básica (radical [8]). A lista dos formatos de documentos inclui XML, HTML/XHTML, PDF, Microsoft Word, PowerPoint, OpenOffice.org, RTF, texto puro (plaintext), Torrent, arquivos de áudio (MP3/MP4, OGG, FLAC, WMA) e muitos outros. Para uma completa lista da instalação atual, veja em *Schema/Parser list*.

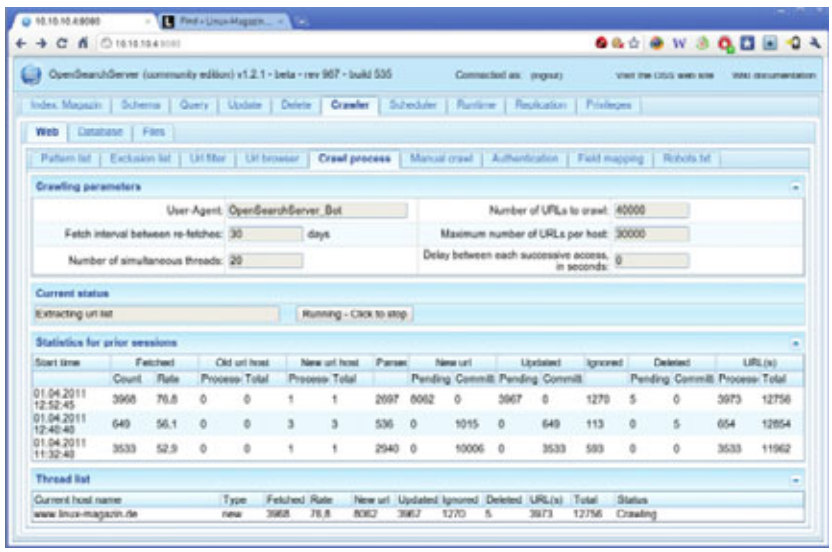

Figura 2 OSS pesquisando o website da Linux Magazine. O número máximo de URLs por campo de host é completamente equivocado. Se você entrar com um valor mais alto aqui, a memória se esgotará rapidamente.

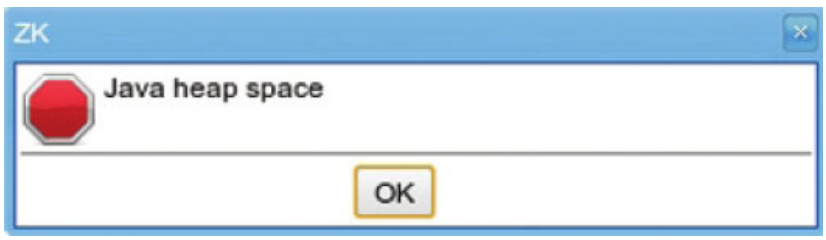

Figura 3 Zkoss apontando um aviso do Java heap space – essa mensagem tipicamente refere-se à falta de memória RAM para o JVM.

Listagem 1: Consulta OSS

```
01 title:($$)^10 OR title:("$$")^10
02 OR
03 titleExact:($$)^10 OR titleExact:("$$")^10
04 OR url:($$)^5 OR
05 url:("$$")^5
06 OR urlSplit:($$)^5 OR urlSplit:("$$")^5
07 OR
08 urlExact:($$)^5
09 OR urlExact:("$$")^5
10 OR content:($$) OR
11 content:("$$")
12 OR
13 contentExact:($$) OR contentExact:("$$")
```

Consulta e lema

Consultas podem ser implementadas como consultas HTTP via interface XML e integradas ao seu website por meio de uma biblioteca de cliente

PHP. A interface de gerenciamento também oferece uma tela de entrada em *Query/Edit/Query*, que pode ser usada para buscas no índice. A **listagem 1** mostra as especificações padrão para a versão anterior, mas que também funciona na 1.2 e é facilmente extensível. A chave de busca é representada por `$$; ^` seguido de um número que define o peso; palavras-chave como `title` ou `urlSplit` definem onde e como exatamente o termo de busca pode ocorrer.

O problema está frequentemente nos detalhes: enquanto o `urlSplit` reduz a URL `http://www.open-search-server.com` para a forma normalizada *http*, *www*, *ope*, *search* e *serv*, o `urlExact` interpreta a mesma URL como os elementos *http*, *www*, *open*, *search* e *server*. De acordo com os desenvolvedores, o OSS usa diferentes analisadores para tal: o analisador de texto identifica grafias similares, enquanto o `urlExact`, `titleExact` e `contentExact` usam o analisador padrão.

Na aba *Scheduler*, você pode criar *cronjobs* para controlar o OSS – por exemplo, para as reconstruções dos índices (`Optimize`). A opção *Runtime* oferece importantes opiniões e comentários a respeito do índice, do cache e dos estágios do sistema. O OSS é também bem equipado para configurações de alta disponibilidade: a aba *Replication* inclui caixas de entrada para as URLs e indexa nomes para instâncias de OSS mais detalhadas.

API e PHP

Os subdiretórios `open-search-server/examples/php/OSSquery` e `open-search-server/php/example` contem alguns exemplos de scripts PHP simples que você pode simplesmente integrar à sua própria página web, retirando, assim, funções de busca da interface de gerenciamento para o navegador do cliente.

Em `webSearch.php`, por exemplo, você pode adicionar seu próprio índice como padrão, o que proporciona uma busca funcional de ponta a ponta, pelo menos parcialmente. Isso porque houve um lado negativo: a interface tem problemas com acentuação em alguns idiomas e não é completamente capaz de implementar os parâmetros da linguagem de buscas Lucene. Realizar consulta na interface gráfica é melhor na maioria das vezes, mas não em todos os casos (**figura5**).

Figura 4 O Open Search Server local está analisando o servidor de web local a menos de 100KBps porque o administrador restringiu ao máximo o número de linhas de execução e frequência de consultas na configuração.

Recomenda-se a API XML. Desenvolvedores podem usá-la para buscas no índice – em quase qualquer linguagem de programação – e mostrar os resultados. Tudo o que você precisa fazer é adicionar os parâmetros necessários nos requerimentos correspondentes de HTTP na forma de um requerimento `GET`:

```
http://localhost:8080/select?
➥use=Index&qt=Query&q=Keyword
```

Quando vê este tipo de consulta, o servidor retorna um arquivo XML contendo as combinações junto com a prévia do texto correspondente (ou *snippets*). Em outras palavra, como um programador, você somente precisa do seguinte:

Figura 5 Se você tentar usar parâmetros mais exóticos para o Lucene diretamente na consulta (por exemplo, a busca com ~), você pode falhar por causa da implementação do OSS.

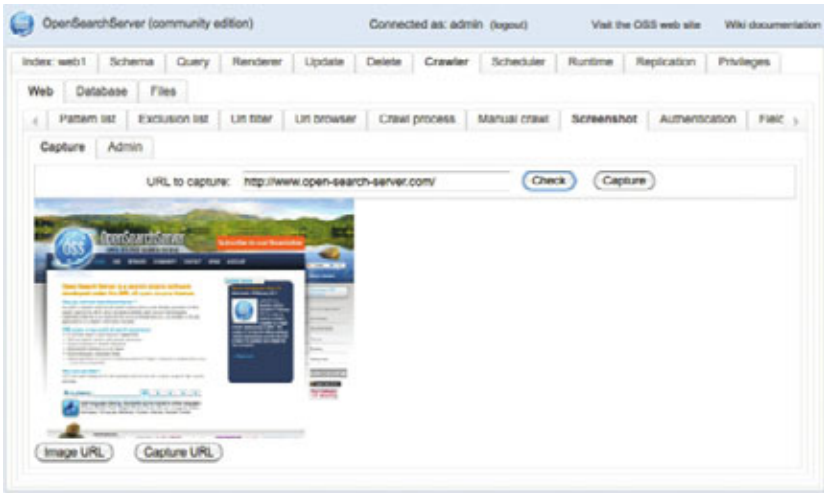

Figura 6 Versões futuras incluem a função screenshot.

- filtrar e validar o *user input* (tome cuidado, a API também pode deletar os documentos do índice!).
- agrupar a *string* para consultar a API.
- analisar o documento XML retornado e formatar a saída.

A **listagem 2** mostra um resumo de um simples script de API. As **linhas 4** e **20** implementam uma proteção mínima contra códigos maliciosos. A **linha 4** proporciona proteção contra scripts de *cross site* (técnica de programação maliciosa), enquanto a **linha 20** remove um `delete` se um ataque for bem sucedido. A matriz associativa `params` coleta todos os parâmetros para a análise de API, sendo que os parâmetros `use`, `qt` e `q` são obrigatórios e todos os outros são opcionais. O template de busca está disponível na distribuição e oferece uma análise bem simples do índice.

O próximo passo é definir quais campos você quer retornar na resposta XML; isso é feito na aba *Query/Returned Fields*. Neste exemplo, os campos são `url` e `host` (**listagem 3**).

Os três parâmetros de colapso especificam se o OSS deve agrupar os resultados. O campo autoritativo é livremente selecionável; este poderia ser o host ou o título da página. Usando esta abordagem, você pode ter associações semelhantes nos resultados até o usuário clicar novamente, melhorando assim a legibilidade.

Na **listagem 2**, este processo acontece da **linha 9** à **11**. No modo `cluster`, a conta máxima é apenas `1`; entretanto, você não pode omitir este parâmetro. A **linha 21** finalmente concatena os parâmetros da busca com a URL do host e codifica os resultados. Um simples chamado ao `simplexml_load_file($this->query)` é o bastante para executar qualquer pedido API formulado deste modo. Você obterá, como retorno, um documento XML como o mostrado na **listagem 3**.

Os resultados finais estão disponíveis como `doc in result`, que interage com um loop `foreach ($hits ->result->doc as $doc)` para cada item. Os dois arquivos modelo PHP da Linux Magazine Online [9] também suportam as características do Lucene, incluindo a aplicação de buscas com coringas como com o caractere `~`, o que antes causava problemas (**figura 5**).

Listagem 2: build-query.php

```
01 function build_query($q){
02   $params=array(); // todos os parâmetros são adicionados aqui e
03   // são removidas as tags
04     $this->q=substr(trim(strip_tags($q),0,80));
05   // basic parameters
06     $params['use']='myIndex'; // índice utilizado
07     $params['qt']='search'; // Query Template
08   // Collapsing
09     $params['collapse.mode']='cluster';
10     $params['collapse.field']='host';
11     $params['collapse.max']=1;
12   // Pagination
13     if (isset($_GET['page'])) $params['start']=((int)
➥ $_GET['page']*10)-9;
14     else $params['start']=1;
15     $params['rows']=$this->rpp;
16     $params['query']=trim($q); // chave de busca
17     $params['sort']='score';
18     $host= 'http://127.0.0.1:8080/select'; // URL do host OSS
19   // In case somebody injected a "Delete" parameter:
20     unset($params['delete']);
21     $this->query=$host.'?'.http_build_query($params);
22 }
```

Novidades na versão 1.2 e previsões

A versão 1.2 apresenta diversas características dignas de aplausos. O buscador web agora diz a você quando ele atualizou a página no índice, usa chaves únicas para lembrar páginas idênticas e permite que você ignore arquivos `robots.txt`. O banco de dados do crawler também entende buscas SQL; você pode desabilitar tanto listas de padrões como de exclusão, e as

análises identificam documentos similares com mais confiabilidade.

A tag `modified` é ainda um problema bastante substancial. No momento, ela detecta se a página foi modificada, mas não detecta se somente a publicidade da página foi substituída ou se o conteúdo em si foi modificado porque a página, por exemplo, contém um feed RSS. Você pode usar o CSS para melhorar esta característica em seus próprios servidores de web usando a tag `div` para esconder menus, propagandas e feeds do indexador:

```
<div class="opensearchserver.
➥ignore"></div>
```

O desenvolvimento está em progresso em relação a muitas características: de acordo com o desenvolvedor Emmanuel Keller, um tipo de page rank (baseado na função `score` do Lucene) foi parcialmente implementado. As funções `score`, `ord`, `rord`, `byte`, `float`, `int`, e `short` estão completas. Um exemplo válido de consulta, então, seria assim: `score() + rord('modified_date')`. A próxima versão incluirá também uma função *screenshot* e uma extensão de renderização. (**figuras 6 e 7**)

O OSS ainda não é perfeito, mas definitivamente está no caminho certo. E é rápido: quando executado em um desktop, respondeu a uma consulta de 25.000 páginas do índice do site da **Linux Magazine** surpreendentemente rápido. Pouquíssimas vezes levou mais de meio segundo, como confirmado pelos nossos testes com centenas de URLs escolhidas aleatoriamente. ■

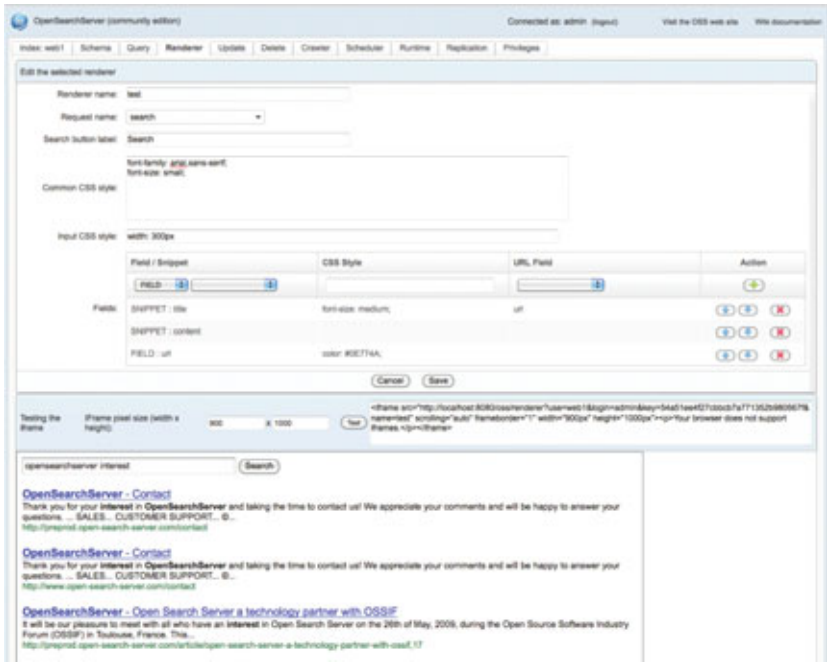

Figura 7 Futuras versões incluem uma extensão de renderização.

Listagem 3: bResposta XML

```
01 <?xml version="1.0" encoding="UTF-8"?>
02   <response>
03   <header>
04     <status>0</status>
05     <query>Query_Name</query>
06   </header>
07   <result name= "response" numFound="111" collapsedDocCount="0"
➥ start="1" rows="10" maxScore="NaN" time="11">
08   <doc score="0.4523" pos="1">
09     <field name= "url">http://www.myserver.com/path/file.html
➥ </field>
10     <field name="host">www.myserver.com</field>
11     <snippet name= "title" highlighted="yes">Page title</snippet>
12     <snippet name="content" highlighted="yes">brief preview of
➥ matches and neighborhood ...</snippet>
13   </doc>
14 </xml>
```

Mais informações

[1] Servidor Open Search: http://www.open-search-server.com/

[2] Tomcat: http://tomcat.apache.org/

[3] Lucene no Apache: http://lucene.apache.org/java/docs/index.html

[4] Agendador Quartz: http://www.quartz-scheduler.org

[5] Zkoss: http://www.zkoss.org/product/zk.dsp

[6] Wiki OSS: http://www.open-search-server.com/wiki/en/index.php/Main_Page

[7] Darkstat: http://dmr.ath.cx/net/darkstat/

[8] Lemma: http://en.wikipedia.org/wiki/Lemma_(linguistics)

[9] Scripts PHP: http://www.linuxmagazine.com.br/issues/80/OSS.zip

Gostou do artigo?

Queremos ouvir sua opinião. Fale conosco em cartas@linuxmagazine.com.br

Este artigo no nosso site: http://lnm.com.br/article/5499